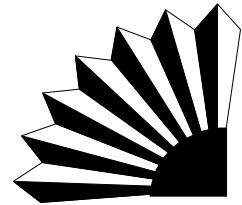


# the Technical Broadcast



Published by the Department of Information Services

Summer 1996

## COBOL/370 and LE/370, Article Five

 by Gary Duffield

**T**his is the fifth in a series of eight articles discussing features of the COBOL/370 programming language.

As mentioned in the first article of this series, COBOL/370 does not have its own unique runtime environment, but shares one with PL/1 and C. This shared runtime environment is known as Language Environment/370 (LE/370).

This article discusses DYNAMIC STORAGE MANAGEMENT, one of COBOL/370's capabilities under LE/370.

Because of its 31-bit addressing capabilities, COBOL/370 can address quite a bit of storage. This has resulted in the creation of some gargantuan (don't you love that word?) working storage tables by programmers looking to gain efficiency by processing 'in memory'. If these tables always have the same number of entries, then 'so be it' . . . they're cool. Unfortunately, in some processes, the

number of entries varies from run to run, and we end up with humongous (another good one) programs that sometimes don't really need all the space they occupy.

"Ah," you say, "that's why I use the OCCURS DEPENDING ON clause in my tables." Yes . . . but, you see, the program still sets aside enough storage to satisfy the TO value you specify on that clause. So even though you may not be using it, the storage is still there.

LE/370 provides CALLABLE SERVICES to programs running under its control. One of these services allows the COBOL/370 programmer to handle these big (okay, so I'm already out of good adjectives) tables much more intelligently.

The strategy is this: instead of defining a table in working storage (and setting aside all that space), let's move the definition to the

*(Continued on page 2)*

### Inside...

**But All I did was Compress the Loadlib . . .**

5-6

**NATURAL WORKFILES**

7

**Year 2000 Update**

8-10

---

# COBOL/370 and LE/370, Article Five

*(Continued from page 1)*

linkage area where it will have no storage associated with it. That puts our program on a crash weight loss plan immediately.

Let's take the table used as an example in the last article and move it to the linkage section:

```
LINKAGE SECTION.  
  01 EMPLOYEE-TABLE.  
    05 NUMBER-OF-EMPLOYEES          PIC S9(4) BINARY.  
    05 EMPLOYEE-RECORD              OCCURS 1 TO 10000 TIMES  
      DEPENDING ON NUMBER-OF-EMPLOYEES.  
      10 NAME                       PIC X(20).  
      10 ID-NO                      PIC 9(9).  
      10 SALARY                     PIC 9(7)V99.
```

Now, because it is in the linkage section, it does not yet define real memory. So we need to invoke LE/370's callable services to get some. But first, we have to figure out how much storage we need. Let's say we want to start out with enough for 5,000 employees:

```
COMPUTE STORAGE-NEEDED =  
  5000 * LENGTH OF EMPLOYEE-RECORD +  
  LENGTH OF NUMBER-OF-EMPLOYEES.
```

Of course, you could do this math at the time you are coding the program and just put the hardcoded value (190002) into the STORAGE-NEEDED field. But imagine the number of elements as a parameter passed to your program instead of a hardcoded 5000. Then the COMPUTE statement above can produce a different answer specific to that run of the program.

Anyway, now that we know how much storage we want, let's go get it:

```
CALL 'CEEGETST' USING HEAP-ID, STORAGE-NEEDED,  
  STORAGE-ADDRESS, FC.
```

All services belonging to LE/370 have names beginning with "CEE." The rest of the name is a sort of mnemonic -- GTST = GeT Storage.

*(Continued on page 3)*

---

# COBOL/370 and LE/370, Article Five

*(Continued from page 2)*

The HEAP-ID (fullword signed integer) should have a value of 0 (zero) to use the default storage heap set up by LE/370 when program execution is begun. We also passed the amount of storage needed (also a fullword signed integer).

We get back a value in storage-address, where the allocated storage can be. But how can we tell if the service was successful? Use the FEEDBACK-CODE (FC) parameter to find this out. A future article will discuss error handling under LE/370 and talk more about the FC. So for now, a value of binary zeros in the 12 character FC is synonymous with success.

The STORAGE-ADDRESS field is a special critter called a POINTER. It is used to pass address locations and is defined in your program like this:

```
05 STORAGE-ADDRESS
```

```
USAGE IS POINTER.
```

That's it!

We now need to connect our description of the table in the linkage section to the actual storage we just allocated. We do that with the COBOL/370 special register called ADDRESS OF. All items defined in the linkage section have this special register connected to them. So:

```
SET ADDRESS OF EMPLOYEE-TABLE TO STORAGE-ADDRESS.
```

And there we have it. A table description that points to enough storage to hold information for 5,000 employees.

Oops! But later our program logic detects that we are about to add the five thousand and first entry. We've used up all the storage! What do we do?

This is cool: LE/370 provides another callable service to resize an existing storage allocation. Suppose we decide to get enough additional storage to hold another 2,500 entries. Since we're resizing, NOT adding on to, we must specify the new TOTAL size. So we would redo our COMPUTE statement above, this time for 7,500 entries (the original 5,000 plus 2,500 more).

*(Continued on page 4)*

---

# COBOL/370 and LE/370, Article Five

(Continued from page 3)

Now that we have our new value for STORAGE-NEEDED, let's resize it:

```
SET STORAGE-ADDRESS TO ADDRESS OF EMPLOYEE-TABLE.  
CALL 'CEECSZT' USING STORAGE-ADDRESS, STORAGE-NEEDED, FC.
```

Perhaps the CZST stands for Change siZe of STorage? In any event, we pass the address of the storage area to be resized. It is important to note that the service MAY ALSO RELOCATE the storage in order to get enough contiguous memory to satisfy your request. So it is necessary to reset our link:

```
SET ADDRESS OF EMPLOYEE-TABLE TO STORAGE-ADDRESS.
```

So there you go—intelligent handling of storage needs from within a COBOL program!

One last thing. It is a good practice to use one last callable service in your termination logic:


```
SET STORAGE-ADDRESS TO ADDRESS OF EMPLOYEE-TABLE.  
CALL 'CEEFRST' USING STORAGE-ADDRESS, FC.
```

You guessed it. FRee the STorage when you're done with it. Sure, it would be automatically freed when your program terminates, but that would raise a condition to LE/370. That's something we probably don't want to do. More on raising a condition in a future article.

More information about DYNAMIC STORAGE MANAGEMENT and other CALLABLE SERVICES can be found in:

IBM SAA AD/Cycle Language Environment/370 Programming Guide (SC26-4818)

The remaining articles of this series will be published in future issues of the *DIS Technical Broadcast*.

If you have any questions about these articles, please contact Gary Duffield at 902-3031. If you would like to obtain a copy of all eight, contact Charie Martin at 902-3112. 

---

# But All I did was Compress the Loadlib...

 by Carol Criscione

**M**any years ago, nearly all Department of Information Services' (DIS) Customer Information Control System (CICS) customer "runtime" programs (modules) were compiled or copied into two B-I-G (HUMONGOUS) shared libraries--one for test and one for production regions. That made things convenient unless the libraries contained corrupted modules or ran out of space thereby impacting all regions using them. The shared libraries grew larger--and larger--and larger. Deleting modules was not a popular option back then as there were security-related issues. Also, customers were requesting more control over their compile procedures and libraries.

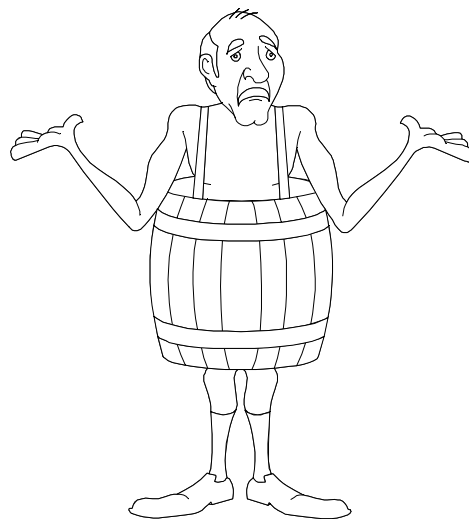
Ultimately, the decision was made to divide the massive runtime libraries into smaller ones which could be maintained and secured by DIS customers. DIS would create and maintain an online selection of CICS-related compile procedures and define the customer libraries to the appropriate CICS regions. The individual agencies would create and maintain their own agency libraries and customized procedures.

The library maintenance staff made the libraries larger when needed and compressed them when the CICS region using them was "down for the night." The security staff designated who would be allowed access to their libraries via RACF security rules.

Of course, it is common knowledge that very little in the data processing arena remains constant long. Over time, many agencies

added a swing shift; some added a graveyard shift. System development activities increased dramatically. Time zones became a factor in CICS availability hours with the practice of telecommuting. Many CICS regions became available 20 or more hours per day and six or more days per week. It seemed like *everyone* wanted the system at the same time. The maintenance window available for CICS-related libraries became smaller--and smaller--and smaller. That presented major challenges for maintenance staff.

CICS-related library maintenance performed when a CICS region is "up" (online) has a high potential of adversely impacting the region, which means a lot of people are unhappy because they can't do their work. The maintenance function *least* desirable when the region is up is the compression function.



(Continued on page 6)

---

# But All I did was Compress the Loadlib...

(Continued from page 5)

In a nutshell (and using a lot of technical license); when CICS comes up, it builds a unique PPT table entry for each program defined to CICS. That table entry contains such information as the program name, the language in which it is written, and where in CICS it can execute. It also reserves a place for the DASD or (hard drive/storage) address of the module. The first time the module is needed, CICS looks up the module name, searches the libraries defined to CICS, finds the DASD address of the module, saves the address in the module PPT table entry for future reference, loads the module residing at that address into CICS “memory,” and starts it.

Here is the problem: when a library is compressed, many programs may be *physically* moved to new locations with new DASD addresses. Unfortunately, any CICS transaction running at the time of the compression will still be referencing the *old* address of the module--now an invalid address. It's similar to moving and not leaving a forwarding address with the U.S. Postal Service.

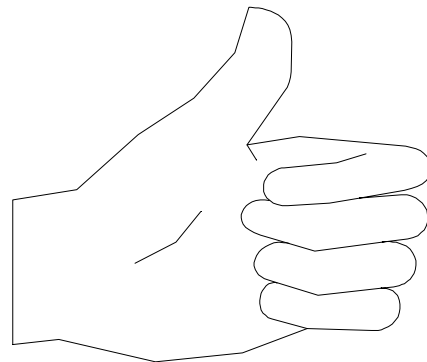
How do programmers change a program used at least once when the CICS is up? The program is changed, compiled, placed into the relevant library, and the CICS **NEWCopy** transaction is used. NEWC replaces the previous address of the module in the PPT table entry with the new address of the changed module. The next time a transaction is started using the module, it references the

new DASD address of the changed module. This event is like moving and *leaving* a forwarding address. Pretty neat, huh?

Why not just compress the library and NEWCopy all the modules executed at least once? Hundreds of programs might need to be “newcopied.” It's unlikely anyone is fast enough to successfully complete that task without a problem! In the end, the affected region might have to be brought down and back up again to clear the problem. That process of “recycling the region” is definitely an undesirable action.

Fortunately, there have been *very* few incidences of CICS problems caused by someone compressing a runtime library when a CICS region is online. The few times it has occurred, a library compression batch job did not run properly or a dynamic compression command was used when the region was up.

“Good job!” to all you talented, hard-working maintenance people who keep the libraries neat, clean, and keep the rest of us working as efficiently as possible. I appreciate it! 📄



---

# NATURAL WORKFILES


 by Tom Thomas

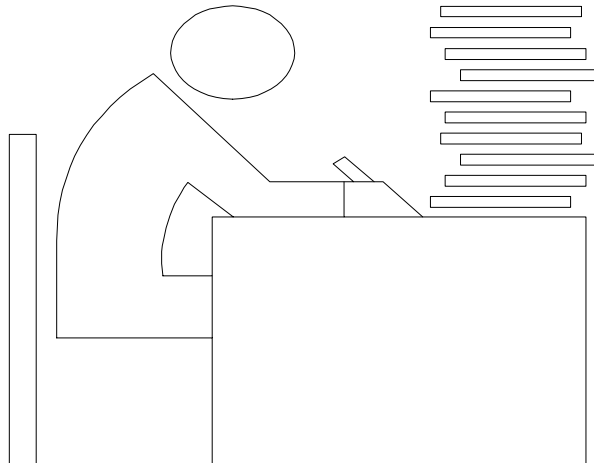
**N**ATURAL version 2.2.7 introduced a new value for the NATURAL WORKBLK profile parameter that may be beneficial to NATURAL programmers who write NATURAL Workfiles. With the NATURAL profile parameter WORKBLK, you can now have the blocksize set automatically by specifying a block size value of "0". This causes the Data Facility System Managed Storage (DFSMS) to set the optimum blocksize automatically. In the example below, SMS set the block size to 27960 (which is half track blocking on disk). If the dataset were going to tape, the blocksize would have been 32K.

Currently, the vendor supplied default value for WORKBLK on all databases is 4628. We plan to make zero the default value for WORKBLK in the near future.

In the interim, you can dynamically override the default value. The following example shows how to do the override. Make sure that the BLKSIZE parameter on the DD statement is set to zero. If you code a BLKSIZE greater than zero, it overrides the NATURAL profile parameter WORKBLK=0 and that is what you get.

```
//TEST      EXEC NAT,DBID=nnn,PARMS=  'WORKBLK=0,STACK=(LOGON *)'  
//*  
//CMWKF01   DD DISP=(NEW,CATLG,DELETE),DSN=TEST.FB40,  
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=40,    BLKSIZE=0),  
//          SPACE=(TRK,(1,1))
```

If you have any questions or wish to discuss this further, please call the DIS CICS/Database Technical Services staff at 902-3045. 



---

# Year 2000 Update

 by Kathy Rosmond

## **T**he Year 2000 Project Overview

The year 2000 poses a challenge to the whole information technology industry at all levels of computing from mainframes to workstations. In an effort to be efficient with data entry and data storage, computer programmers stored the year as two digits (96) instead of four (1996). When the next century arrives, computers will interpret the date as 1900 and calculate inaccurate results when performing comparisons on dates, arithmetic operations, or sorting by date. Major conversions must be done. Along with the conversions, identifying links to other computer programs and verifying that the results are accurate will be an enormous task. It is predicted by the Gartner Group that only 25 percent of critical state government computer systems will be ready to handle it.

In an effort to meet the challenge, the Department of Information Services (DIS) assigned project managers, Stan Ditterline and Kathy Rosmond, to facilitate statewide efforts. The team takes a phased approach to the plan. In Phase I, the project team will assist agencies by providing the tools and services needed to develop credible estimates for planning and budgeting. In Phase II, agencies will prepare detailed analyses and a plan for conversions and testing. DIS will provide technical tools, project management, and support to convert and test critical agency applications. Phase III will be the actual conversions and testing. This must be

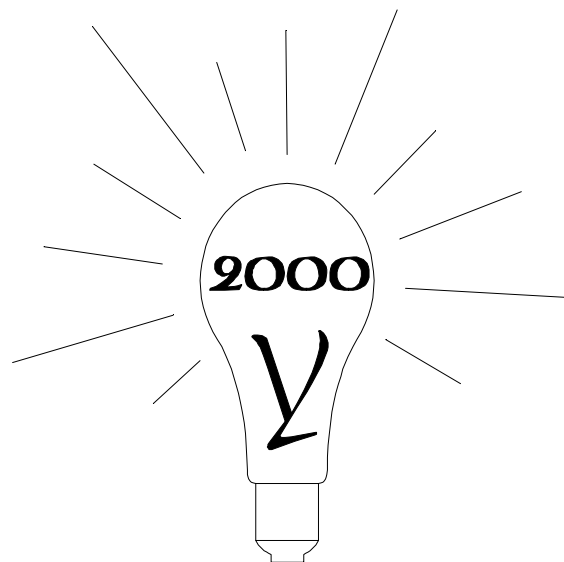
completed before the year 2000. There is no way to turn back the clock!

A Year 2000 Special Interest Group (SIG) has formed and meets the third Wednesday of each month. The SIG works with DIS to identify needed state government technical resources and participates in evaluating resulting statewide contracts. The SIG also provides a forum for sharing information and experiences. An executive steering committee made up of deputy directors provides high-level perspective and support for the project.

With agencies planning now and working together, Washington State computer systems will be prepared for the next century.

### **DIS Year 2000 WWW Home Page**

The Department of Information Services' year 2000 team recently introduced its



*(Continued on page 9)*



---

# Year 2000 Update

*(Continued from page 8)*

project on the World Wide Web (WWW) through the Washington State home page. Though the site is newly constructed, there is already valuable information about the year 2000 at your fingertips. The address of the site for the Project 2000 Resource Information Center is:  
<http://www.wa.gov/dis/2000/y2000.htm>.

Contents include:

- year 2000 Calendar of Events
- general information for the technical and nontechnical
- year 2000 list serve excerpts
- links to other sites of interest
- acquisitions and contracts supporting government year 2000 projects
- year 2000 hardware and software compliance surveys

The site has links to other sites like Peter de Jager's Year 2000 Information Center. Information specific to Washington State year 2000 efforts can be found under the year 2000 Calendar of Events. Selected excerpts from Peter de Jager's year 2000 list serve will be stored for a three month period. Other states' and corporations' year 2000 experiences may be available by visiting "Links to Other Sites of Interest."

The content of the WWW page will grow as agencies and companies continue their planning, conversions, and testing for the year 2000, producing a valuable, easy to use,

and timely resource for people involved in this project.

If you have suggestions for the DIS Year 2000 Information Resource Center home page, please contact Judy Politz at (360) 902-3046 or email at [judyp@dis.wa.gov](mailto:judyp@dis.wa.gov).

## **Year 2000 Analysis and Planning Guideline**

The Department of Information Services and the Year 2000 Special Interest Group worked to develop the Year 2000 Planning and Analysis Guideline to assist agencies developing high-level time, cost, and resource estimates for planning and budgeting for year 2000 conversion efforts. Since many projects require a coordinated effort between state agencies, it is essential to gather consistent data. The guideline contains a matrix to determine conversion risks and business impact on essential state services.

A copy of the guideline is available at the DIS Year 2000 Information Resource Center home page at [http://www.wa.gov/dis/2000/3\\_info.htm](http://www.wa.gov/dis/2000/3_info.htm) or by contacting Kathy Rosmond at (360) 902-3445 or email at [kathyr@dis.wa.gov](mailto:kathyr@dis.wa.gov).

**WA Year 2000 Special Interest Group (SIG)** Meetings; third Wednesday of the month; Contact: Stan Ditterline, DIS, (360) 902-3574 or email at [stand@dis.wa.gov](mailto:stand@dis.wa.gov).

*(Continued on page 10)*

---

# Year 2000 Update

*(Continued from page 9)*

## **ADABAS Natural Agreement Negotiated**


The Department of Information Services negotiated a limited use master agreement for year 2000 date analysis for Software AG ADABAS Natural language programs. The apparent successful contractor is Data Dimensions Inc., 777 108 Avenue NE, Suite 2070, Bellevue, WA 98004.

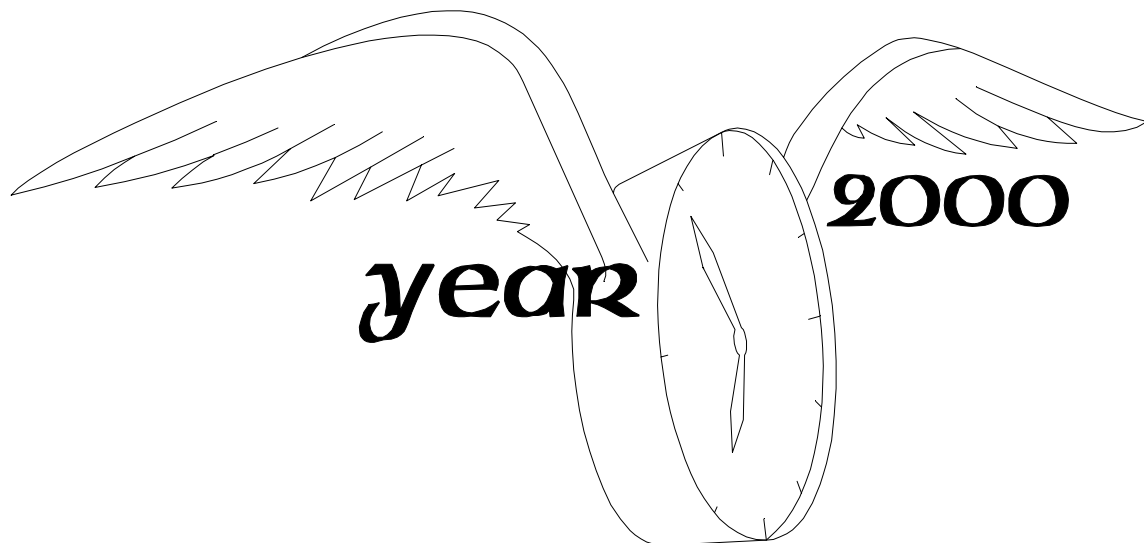
The agreement allows agencies to buy services under this contract by executing a

statement of work directly with Data Dimensions.

For information on this agreement, contact Becci Riley at (360) 902-3509 or visit the DIS Year 2000 Project Information Resource Center at <http://www.wa.gov/dis/2000/y2000.htm>.

### **For More Info**

Please contact Kathy Rosmond at (360) 902-3445 or email at [kathyr@dis.wa.gov](mailto:kathyr@dis.wa.gov) for further information on the year 2000. 



# the Technical Broadcast



Charie L. Martin, Editor  
Department of Information Services  
Adams Building  
1310 Jefferson ST SE  
Mail Stop: 42445  
Olympia, WA 98504-2445

***Technical Broadcast Staff*** - The *Technical Broadcast* is published quarterly by the Department of Information Services (DIS). The purpose is to provide a forum for customer information-sharing of upcoming system enhancements and optimization tips. We invite your articles, comments, and suggestions.

DIS is an equal opportunity employer and does not discriminate on the basis of race, religion, color, sex, age, nationality, or disability.



Washington State Department of  
***Information Services***